**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
ON APPEAL FROM THE EXAMINER TO THE BOARD
OF PATENT APPEALS AND INTERFERENCES**

In re Application of:	Rikin S. Patel
Serial No.:	10/729,607
Filed:	December 5, 2003
Group Art Unit:	2451
Examiner:	Glenford J. Madamba
Confirmation No.:	2892
Title:	SYSTEM AND METHOD FOR FAULT MANAGEMENT IN A SERVICE-ORIENTED ARCHITECTURE

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

Dear Sir:

APPEAL BRIEF

Appellant has appealed to the Board of Patent Appeals and Interferences ("*Board*") from the Final Office Action dated May 16, 2008 ("*Final Office Action*"). Appellant filed a Notice of Appeal and Pre-Appeal Brief on August 12, 2008 with the statutory fee of $510.00. This Appeal Brief is filed in response to Notice of Panel Decision from Pre-Appeal Brief Review dated October 23, 2008, finally rejecting Claims 2-12 and 15-45 pending in this application.

Real Party In Interest

This Application is currently owned by ELECTRONIC DATA SYSTEMS CORPORATION as indicated by:

an assignment recorded on 12/05/2003 from inventor Rikin S. Patel, to Electronic Data Systems Corporation, in the Assignment Records of the PTO at Reel 014772, Frame 0930 (3 pages).

Related Appeals and Interferences

To the knowledge of Appellant's counsel, there are no known interferences or judicial proceedings that will directly affect or be directly affected by or have a bearing on the Board's decision regarding this Appeal.

Status of Claims

Claims 2-12 and 15-45 are pending and stand rejected pursuant to a Final Office Action dated May 16, 2008 ("*Final Office Action*") and a Notice of Panel Decision from Pre-Appeal Brief Review dated October 23, 2008 ("*Panel Decision*"). Specifically, Claims 2-12 and 15-45 are rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 7,146,544 issued to Hsu et al. ("*Hsu*") in view of U.S. Patent Application Publication No. 2005/0015472 issued to Catania et al. ("*Catania*"). For the reasons discussed below, Appellant respectfully submits that the rejections of Claims 2-12 and 15-45 are improper and should be reversed by the Board. Accordingly, Appellant presents Claims 2-1 and 15-45 for Appeal. All pending claims are shown in Appendix A, attached hereto.

Status of Amendments

All amendments submitted by Appellant have been entered by the Examiner.

Summary of Claimed Subject Matter

FIGURE 1 illustrates a method for managing faults in a service-oriented architecture. At step 110, service is invoked by a service consumer. The service consumer may be internal to the system architecture, or may be an external consumer accessing the system architecture via a network. The service consumer preferably invokes the service at step 110 utilizing a web service description language (WSDL). Continuing with step 110, when a service provider receives the service request it preferably begins to fulfill the service request invoked by the service consumer. At step 120 the system may encounter a fault or exception. If at step 120 there is no fault or exception encountered, at step 122 the service is completed according to the service request by the service consumer at step 110. If, at step 120, a fault or exception is encountered, at step 130 that fault is persisted through the system. Fault persistence may include labeling the fault with a unique identifier such as a fault number, or letter number combination, for persistence through the system. At step 130, the fault is preferably persisted into a persistent data store such as a database that would preferably be coupled to a network, but in any event will be at least coupled to the system tracking the faults within the service-oriented architecture. In any given embodiment, the persistent store may be dedicated or integrated with other data storage requirements of the architecture. (Page 6, lines 5-22.)

At step 140, the system determines whether a response is required for the fault that has been persisted at step 130. (Page 6, lines 23-24.)

If the service invoked at step 110 requires a response upon being fulfilled, the response would include the fault information in addition to any of the service information that is provided other than the information that caused the fault to occur within the system. If, for example, at step 140 no response is required, the system may continue to persist the fault at step 142 until a response is required, if ever. An example of when a fault response may not be required immediately upon discovering the fault could occur when the service consumer invoking the service at step 110 is a system or subsystem within the service-oriented architecture itself. Additionally, a fault response may not be required until requested by a fault service consumer. Fault service consumers may be any entity, individual or system, either internal or external, requesting fault information. Accordingly, at any later date a user may be able to access the architecture to determine where any or all of the faults occurring in this service-oriented architecture originate from, or what the cause of the faults might be attributed to, in order to improve the service provided by the service-oriented architecture or to repair systems that have failed or consistently fail within the service-oriented architecture. (Page 6, line 25 - page 7, line 8.)

The system may implement the fault management service at step 150. The fault management service may include translating the fault information to or from a WSDL, querying the persistent store to determine the source of faults from a particular sub-system within the service-oriented architecture, or any other fault management service required by the system-architecture, or the service request invoked by the service consumer at step 110. For example, a fault response may be automatically required to be given to the service consumer. Alternatively, a fault response may be provided to an internal system such as a repair system, or fault management tracking system, that allows the system to identify problems within the system architecture or systems within the architecture that consistently fail. Additionally, the fault management service invoked at step 150 may be operable to format a response in a WSDL to be sent to another entity such as the service consumer, or a fault service consumer if the fault service consumer is different from the service consumer. A fault service consumer may be different from the service consumer in any number of situations. For example, the fault service consumer may be the original service consumer, a unique fault service consumer, an internal sub-system, an automated function of the architecture, or any other requestor of fault information. Step 150 may also be accomplished by sending a WSDL-formatted response to other systems within the service-oriented architecture. In yet another embodiment, sending results at step 160 may include transmitting a fault response translated into a computer language or protocol requested by the service consumer in the invocation of service at step 110. (Page 7, lines 9-30.)

FIGURE 2 illustrates a system 200 for managing faults within a service-oriented architecture. System 200 may include one or more service consumers 210 coupled to a network 220. Service consumers 210 may be individuals, businesses, or other entities desiring services from a service-oriented architecture. Architecture 230 is preferably coupled to network 220 and may include the following components: business service interface 240, service implementation 250, persistent store 260, fault service implementation 270, fault service interface 280, and fault network 290. Fault network 290 may be a LAN, MAN, WAN, wireless, wireline, optical, or, in the case of a system architecture spread over larger distances, a portion of the Internet. Additionally, fault network 290 may be unnecessary if other components present in the system architecture are coupled to network 220, which may be the Internet, the World Wide Web, intranet, LAN, MAN, WAN, wireline, wireless, optical, or any other type of network. Additionally, a fault service consumer 212 may be coupled to network 220, or may be an internal subsystem of architecture 230. (Page 7, line 31 - page 8, line 12.)

In one embodiment, a service consumer 210 invokes a service via network 220 from architecture 230. Architecture 230 may be any type of service provider, such as a business, corporation, group of businesses, affiliated businesses, or government entities, or any other service-oriented architecture capable of providing services over a loosely coupled integrated network. Upon invoking a service by architecture 230, the service request by service consumer 210 is preferably directed to the business service interface 240. In one embodiment, the service request is received by the business service interface in a web services description language (WSDL), such as remote procedure call (RPC), hypertext transfer protocol (HTTP), Java message service (JMS) or any other WSDL that is registered in a universal description discovery and integration (UDDI) registry . However, in other embodiments, the service request from service consumer 210 may be transmitted in a proprietary language used by architecture 230 or in any language capable of interpretation by architecture 230. (Page 8, lines 13-25.)

Upon receiving the service request, business service interface 240 preferably translates the message, if necessary, from the WSDL into the operating protocol or language of the service architecture 230. However, system architecture 230 may also be configured to operate within a WSDL, thereby obviating the need for business service interface 240 to translate the service request from a WSDL. Additionally, business service interface 240 may be operable to translate the service request into a WSDL other than the WSDL used by the service consumer 210 to transmit the original service request. For example, the service request may be received as an HTTP and translated into an RPC or a JMS, among others. In such a case, architecture 230 may operate solely in a WSDL thereby streamlining the service response and service management by reducing the need to translate between different computer or network languages and protocols. (Page 8, line 26 - page 9, line 5.)

Once business service interface 240 has translated and/or processed the service request from service consumer 210, it may transmit the request to service implementation 250 via fault network 290. In one embodiment, service implementation 250 may be resident on the same server, computer system, or other system, thus enabling the service request to be accessed by service implementation 250 without transmitting the request via fault network 290. Once service implementation 250 receives the service request, it directs the service request to the appropriate subsystem coupled to architecture 230 to fulfill the service request (not explicitly shown). As an example only, and not by way of limitation, service consumer 210 may request a price quote on a consumer item. Upon receiving the service request, system architecture 230 may direct the service request to a consumer pricing system coupled to architecture 230 in order to fulfill the service request. Upon receiving the request, the consumer pricing service subsystem preferably responds with the information requested to architecture 230. However, due to any number of possibilities, a fault may occur in either the format of the request by the service consumer, the system required to provide the consumer price by the consumer pricing subsystem, the inability of the system architecture 230 to properly transmit the service request, or any other type of error which prevents the fulfillment of the service request. (Page 9, lines 6-24.)

Upon determining that an error, fault, or exception has occurred in the fulfillment of the service request, service implementation 250 preferably persists the fault by directing the fault information to be stored in persistent store 260. In addition to directing the storage of the fault information in persistent store 260, service implementation 250 may label the fault information with a unique identifier that makes accessing fault information easier to accomplish. Labeling each individual fault response with a unique identifier may allow the fault responses to be categorized in any number of ways. For example, the fault information may be categorized by the source of the fault, through either a sub-system or network location, such as an internet protocol (IP) address, or it may be stored by the type of fault, such as a request time-out, a system, or sub-system failure, or any combination of fault sources or locations that assist service implementation to effectively catalog the fault information in persistent store 260. (Page 9, line 25 - page 10, line 5.)

Architecture 230 may include one or more fault service interfaces 280. Fault service interface 280 is preferably operable to translate any fault responses, if required, for transmission to a fault service consumer 212. Alternatively, if architecture 230 operates solely in a WSDL format or protocol, the translation, and/or transmission functions performed by fault service interface 280 may be performed by fault service implementation 270. Additionally, fault service consumer 212 may be an entity located within architecture 230, or may be an entity or system located outside of architecture 230, such as a service consumer 210.

Fault service consumer 212 may also be any other entity requesting fault response information from architecture 230, in order to improve service of the service-oriented architecture, or to maintain or track faults occurring within architecture 230. (Page 10, lines 6-16.)

FIGURE 3 illustrates a system 300 for fault management for a service-oriented architecture. System 300 may include components of system architecture 302 having one or more operator terminals 310, a data management system 320, one or more function modules 330, and a database or persistent store 340. System architecture 302 may also have other components not illustrated in FIGURE 3. The various components may be located at a single site or, alternatively, at a number of different sites. The components of system architecture 302 may be coupled to each other using one or more links, each of which may include one or more computer busses, local area networks (LANs), metropolitan area networks (MANs), wide area networks (WANs), portions of the Internet, or any other appropriate wireline, optical, wireless, or other links. An operator terminal 310 may provide an operator access to data management system 320 to configure, manage, or otherwise interact with data management system 320. An operator terminal 310 may include a computer system (which may include one or more suitable input devices, output devices, processors and associated memory, mass storage media, communication interfaces, and other suitable components) or other suitable device. (Page 10, lines 17-32.)

A function module 330 may provide particular functionality associated with handling service requests or handling transactions in a web service description language (WSDL) according to system 300. As an example only, and not by way of limitation, a function module 330 may provide functionality associated with web services, fault persistence, and diagnostics. A function module 330 may be called by data management system 320 (possibly as a result of data received from an operator terminal 310), a service consumer 392, an internal service consumer 384, a fault service consumer 386, or an internal system 382, and, in response, provide the particular functionality associated with function module 330. A function module 330 may then communicate one or more results to data processing unit 350 or one or more other suitable components of data management system 320, which may use the communicated results to create, modify, or delete one or more data files associated with one or more service requests, provide data to an operator at an operator terminal 310, or a service consumer 392, a fault service consumer 386, an internal system 382, or an internal service consumer 384 or perform any other suitable task. Function modules 330 may be physically distributed such that each function module 330, or multiple instances of each function module 330, may be located in a different physical location geographically remote from each other and/or from data management system 320. In the embodiment shown in FIGURE 3, function modules 330 include a web services module 332, a fault persistence module 334, and a diagnostic module 336. (Page 12, lines 3-22.)

As the service request is being fulfilled, various system exceptions, or faults, may be encountered. Upon encountering an exception in the execution of the service request, fault persistence module 334 preferably directs the information regarding the fault to be stored in database 340, which also may be referred to as a persistent store 340. In addition to storing fault information, fault persistence module 334 preferably labels the fault information for each service request with a unique identifier that enables the fault information to be readily retrieved from persistent store 340 by data management system 320. A diagnostic module 336 may be present in system architecture 302 that identifies the source of any exceptions encountered during fulfillment of the service request, or, alternatively, to identify the location of any subsystems or internal systems within system architecture 302 that caused an exception, or any system coupled to system architecture 302 that was the source of the exception. (Page 13, lines 13-25.)

In addition to directing data management system 320 to store fault information in persistent store 340, fault persistence module 334 may also be operable to fulfill fault service requests from a fault service consumer 386. Fault service consumer 386 may be an internal system 382 coupled to sub-network 380, a service consumer 392 accessing system architecture 302 via communications network 390, or an internal service consumer 384 accessing data management system 320 for a particular service. Upon receiving a fault service request, fault persistence module 334 preferably retrieves the fault information from the service request that it contained the exception from persistent store 340. Additionally, web services module 332 may be called upon by data management system 320 to translate the fault information into a WSDL, or other language readable by fault service consumer 386. Alternatively, fault service consumer 386 may be a dedicated fault system that catalogs and tracks all faults located or occurring within system architecture 302 or any system coupled thereto. In such a case, fault service consumer 386 may include a console or other computer generated display that allows the tracking and servicing of information generated through fault persistence in architecture 302. (Page 13, line 26 - page 14, line 9.)

Preferably, fault persistence module 334 is operable to selectively incorporate different aspects of fault information retrieved from persistent store 340 according to the specifications of system architecture 302. For example, the administrator of system architecture 302 may desire to provide different information to different fault service consumers based on the association of the fault service consumer with the service provider administering system architecture 302. Thus, for a fault service consumer located within system architecture 302, more information may be provided than to an external fault service consumer such as an auditor or a service consumer 392 acting as a fault service consumer. (Page 14, lines 10-18.)

With regard to the independent claims currently under Appeal, Appellants provide the following concise explanation of the subject matter recited in the claim elements. For brevity, Appellants do not necessarily identify every portion of the Specification and drawings relevant to the recited claim elements. Additionally, this explanation should not be used to limit Appellants' claims but is intended to assist the Board in considering the Appeal of this Application.

For example, independent Claim 2 recites:

> A method for managing faults in a web service architecture (e.g., Figure 1, reference numerals 100-160; Page 6, line 5 through Page 7, line 30) comprising:
>
> receiving a service request in a web service language, wherein the service request comprises invoking a service over a network (e.g., Figure 1, reference numeral 110; Figure 2, reference numerals 210-230; Figure 3, reference numeral 320; Page 6, lines 6-11; Page 8, lines 13-19; Page 11, lines 1-6);
>
> translating the service request into a non-web service language (e.g., Figure 2, reference numeral 240; Page 8, line 19 through Page 9, line 5);
>
> executing the service request (e.g., Figure 1, reference numeral 110; Figure 2, reference numeral 250; Figure 3, reference numeral 320; Page 6, lines 10-11; Page 9, lines 6-17; Page 11, lines 1-6);
>
> encountering an exception during the execution, wherein the execution comprises a fault preventing the fulfillment of the service request (e.g., Figure 1, reference numeral 120; Figure 2, reference numeral 250; Figure 3, reference numeral 320; Page 6, lines 11-14; Page 9, lines 18-24; Page 11, lines 1-6);
>
> persisting the fault (e.g., Figure 1, reference numeral 130; Figure 2, reference numerals 250-260; Figure 3, reference numeral 320; Page 6, lines 14-22; Page 9, line 25 through Page 10, line 5; Page 11, lines 1-6); and
>
> providing a fault response (e.g., Figure 1, reference numeral 140; Figure 2, reference numeral 280; Figure 3, reference numeral 320; Page 6, line 23 through Page 7, line 30; Page 10, lines 6-16; Page 11, lines 1-6).

As another example, Claim 11 recites:

A system for managing faults in a service-oriented architecture (e.g., Figure 2, reference numeral 200; Figure 3, reference numeral 300; Page 7, line 31 through Page 10, line 16) comprising:

a service interface (e.g., Figure 2, reference numeral 240; Page 8, lines 19-25) operable to:

receive a service request via a network, the service request received in a web service language (e.g., Figure 1, reference numeral 110; Figure 2, reference numeral 240; Page 6, lines 6-11; Page 8, lines 19-25); and

translate the service request into a non-web service language (e.g., Figure 2, reference numeral 240; Page 8, line 19 through Page 9, line 5);

a service implementation coupled to the service interface, the service implementation operable to perform the service request and determine the source of any fault encountered in the performance (e.g., Figure 1, reference numeral 120; Figure 2, reference numeral 250; Page 6, lines 11-14; Page 9, lines 6-24; Page 11, lines 1-6);

a persistent store operable to persist any faults encountered in the performance (e.g., Figure 1, reference numeral 130; Figure 2, reference numerals 250-260; Figure 3, reference numeral 320; Page 6, lines 14-22; Page 9, line 25 through Page 10, line 5); and

a fault service interface operable to transmit fault information (e.g., Figure 1, reference numeral 140; Figure 2, reference numeral 280; Page 6, line 23 through Page 7, line 30; Page 10, lines 6-16).

As another example, Claim 26 recites:

A system for managing faults in a web service architecture (e.g., Figure 3, reference numeral 300; Page 7, line 31 through Page 10, line 16) comprising:

a web service module coupled to a network and operable to manage service requests in a web service language (e.g., Figure 3, reference numeral 330; Page 12, line 3 through Page 13, line 12), the web service module operable to:

receive a service request via a network, the service request received in the web service language (e.g., Figure 3, reference numeral 330; Page 12, lines 3-30); and

translate the service request into a non-web service language (e.g., Figure 3, reference numeral 330; Page 12, line 30 through Page 13, line 12);

a diagnostic module operable to fulfill the service request and identify faults associated with the service request (e.g., Figure 3, reference numeral 336; Page 13, lines 20-25); and

a fault persistence module operable to store the faults in a persistent store (e.g., Figure 3, reference numeral 334; Page 13, line 13 through Page 14, line 18).

As another example, Claim 45 recites:

A system for managing faults in a web services architecture (e.g., Figure 2, reference numeral 200; Figure 3, reference numeral 300; Page 7, line 31 through Page 10, line 16) comprising:

a system interface operable to receive a service request in a web services format (e.g., Figure 1, reference numeral 110; Figure 2, reference numeral 240; Page 6, lines 6-11; Page 8, lines 19-25), the system interface further operable to translate the service request into a non-web service format (e.g., Figure 2, reference numeral 240; Page 8, line 19 through Page 9, line 5);

a service implementation operable to fulfill the service request, generate a fault report, and persist the fault, the persistence comprising storing the fault report in a persistent store, wherein generating a fault report comprises detecting a fault during the fulfillment of the service request, and persisting the fault comprises attaching a unique identifier to the fault report (e.g., Figure 1, reference numerals 120-130; Figure 2, reference numerals 250-260; Page 6, lines 11-22; Page 9, line 6 through Page 10, line 5; Page 11, lines 1-6);

a fault service implementation operable to retrieve the fault report from the persistent store and translate the fault report into a web service format (e.g., Figure 1, reference numeral 140; Figure 2, reference numeral 270; Page 6, line 23 through Page 7, line 30); and

a fault service interface operable to receive fault service requests and transmit a fault service response (e.g., Figure 1, reference numeral 140; Figure 2, reference numeral 280; Page 6, line 23 through Page 7, line 30; Page 10, lines 6-16).

Grounds of Rejection to be Reviewed on Appeal

Are Claims 2-12 and 15-45 unpatentable under 35 U.S.C. § 103(a) over U.S. Patent No. 7,146,544 issued to Hsu et al. ("*Hsu*") in view of U.S. Patent Application Publication No. 2005/0015472 issued to Catania et al. ("*Catania*")?

Arguments

The Examiner rejects Claims 2-12 and 15-45 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 7,146,544 issued to Hsu et al. ("*Hsu*") in view of U.S. Patent Application Publication No. 2005/0015472 issued to Catania et al. ("*Catania*"). For at least the following reasons, Appellants respectfully submit that these rejections are improper and should be reversed by the Board. Appellants address independent Claims 2, 11, 26, and 45 and dependent Claims 6, 21-22, 24, and 39-40 below.

I. Legal Standard for Obviousness

The question raised under 35 U.S.C. § 103 is whether the prior art taken as a whole would suggest the claimed invention taken as a whole to one of ordinary skill in the art at the time of the invention. One of the three basic criteria that must be established by an Examiner to establish a *prima facie* case of obviousness is that "the prior art reference (or references when combined) must teach or suggest ***all the claim limitations***." *See* M.P.E.P. § 706.02(j) citing *In re Vaeck*, 947 F.2d 488, 20 U.S.P.Q.2d 1438 (Fed. Cir. 1991) (emphasis added). "***All words*** in a claim must be considered in judging the patentability of that claim against the prior art." *See* M.P.E.P. § 2143.03 citing *In re Wilson*, 424 F.2d 1382, 1385 165 U.S.P.Q. 494, 496 (C.C.P.A. 1970) (emphasis added).

In addition, even if all elements of a claim are disclosed in various prior art references, which is certainly not the case here as discussed below, the claimed invention taken as a whole still cannot be said to be obvious without some reason why one of ordinary skill at the time of the invention would have been prompted to modify the teachings of a reference or combine the teachings of multiple references to arrive at the claimed invention.

The controlling case law, rules, and guidelines repeatedly warn against using an Appellant's disclosure as a blueprint to reconstruct the claimed invention. For example, the M.P.E.P. states, "The tendency to resort to 'hindsight' based upon Appellant's disclosure is often difficult to avoid due to the very nature of the examination process. However, impermissible hindsight must be avoided and the legal conclusion must be reached on the basis of the facts gleaned from the prior art." M.P.E.P. § 2142.

The U.S. Supreme Court's decision in *KSR Int'l Co. v. Teleflex, Inc.* reiterated the requirement that Examiners provide an explanation as to why the claimed invention would have been obvious. *KSR Int'l Co. v. Teleflex, Inc.*, 127 S.Ct. 1727 (2007). The analysis regarding an apparent reason to combine the known elements in the fashion claimed in the patent at issue "should be made explicit." *KSR*, 127 S.Ct. at 1740-41. "Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness." *Id.* at 1741 (internal quotations omitted).

The new examination guidelines issued by the PTO in response to the *KSR* decision further emphasize the importance of an explicit, articulated reason why the claimed invention is obvious. Those guidelines state, in part, that "[t]he key to supporting any rejection under 35 U.S.C. 103 is the clear articulation of the reason(s) why the claimed invention would have been obvious. The Supreme Court in *KSR* noted that the analysis supporting a rejection under 35 U.S.C. 103 should be made explicit." *Examination Guidelines for Determining Obviousness Under 35 U.S.C. 103 in View of the Supreme Court Decision in KSR International Co. v. Teleflex Inc.*, 72 Fed. Reg. 57526, 57528-29 (Oct. 10, 2007) (internal citations omitted). The guidelines further describe a number of rationales that, in the PTO's view, can support a finding of obviousness. *Id.* at 57529-34. The guidelines set forth a number of particular findings of fact that must be made and explained by the Examiner to support a finding of obviousness based on one of those rationales. *See id.*

II. Claim 45 is Allowable over the Proposed *Hsu-Catania* Combination

Independent Claim 45 recites:

A system for managing faults in a web services architecture comprising:

a system interface operable to receive a service request in a web services format, *the system interface further operable to translate the service request into a non-web service format*;

a service implementation operable to fulfill the service request, generate a fault report, and persist the fault, the persistence comprising storing the fault report in a persistent store, wherein generating a fault report comprises detecting a fault during the fulfillment of the service request, and *persisting the fault comprises attaching a unique identifier to the fault report*;

a fault service implementation operable to retrieve the fault report from the persistent store and *translate the fault report into a web service format*; and

a fault service interface operable to receive fault service requests and transmit a fault service response.

Because the proposed *Hsu-Catania* combination does not disclose, teach, or suggest at least the claim elements emphasized above, Appellants submit that Claim 45 is allowable over the proposed *Hsu-Catania* combination.

A. The *Hsu-Catania* combination does not disclose, teach, or suggest a system interface operable to *"translate the service request into a non-web service format"*

The rejection of Claim 45 is deficient at least because the *Hsu-Catania* combination does not disclose, teach, or suggest a system interface operable to "translate the service request into a non-web service format." In the *Final Office Action*, the Examiner acknowledges that *Hsu* fails to disclose the recited claim features and instead relies on *Catania*. Appellants respectfully disagree.

Catania discloses "three primary roles: service provider, service registry, and service requester." (*Catania*, page 1, paragraph 7). "The service provider is the entity that provides access to the Web service and **publishes the service description in a service registry**." (*Catania*, page 1, paragraph 7, emphasis added). By contrast, **"[t]he service requestor finds the service description in a service registry** or other location and can use the information in the description to bind to a service." (*Catania*, page 1, paragraph 7, emphasis added). With regard to the messages that are sent, *Catania* discloses that "[w]eb services typically send XML messages formatted in accordance with the Simple Object Access Protocol (SOAP) specification." (*Catania*, page 1, paragraphs 8). *Catania* further clarifies:

> The XML messages are described using the Web Services Description Language (WSDL) specification, which, along with the Universal Description Discovery and Integration (UDDI) registry, provides a definition of the interface to a Web service and identifies service providers in a network. The WSDL specification is an XML-based language used to define Web services and describe how to access them. **An application trying to use a particular Web Service can often use WSDL to find the location of the Web service, the function calls available, and the format that must be followed to access the Web service. Therefore, the client first obtains a copy of the WSDL file from the server and then uses the information in this file to format a SOAP request.**

(*Catania*, page 1, paragraphs 8-9, emphasis added). Thus, *Catania* merely discloses that web service requests are sent in the WSDL format. The service requestor must obtain a copy of the

WSDL file from the server and then format the request in the proper SOAP request format prior to it being sent. Because the SOAP format is a web service format,[1] Appellant contends that using the WSDL registry to format the request in a SOAP format does not include "translat[ing] the service request into a **non-web service format**." Furthermore, *Catania* explicitly describes that such formatting is done prior to the message being sent. Accordingly, there is no disclosure in *Catania* of a system interface that is operable to receive the service request and then "translate the service request into a non-web service format," as recited in Claim 45.

Appellants note the PTO's reliance on paragraphs 68-70 of *Catania* in the *Final Office Action*. However, the cited portion of *Catania* merely discloses a distributed business processes example in which an auction manager "offers a management service that monitors the progress of" a request for quotes (RFQ) process 510. (*Catania*, page 5, paragraph 69). "In one embodiment, RFQ process 510 is implemented in the Business Processes Execution Language (BPEL)," which "is an XML-based language designed to enable task sharing for a distributed computing environment, even across multiple organizations, using a combination of Web services. (*Catania*, page 5, paragraph 69). Thus, the service requestor (Companies C2, C3, C4) transmits requests to the web server (auction manager 500) in BPEL. There is no disclosure that the auction manger 500 is operable to receive the service request and then "translate the service request into a non-web service format," as recited in Claim 45. To the contrary, *Catania* specifically states that the auction manager implements the process in BPEL. Thus, no translation occurs.

For at least these reasons, the *Hsu-Catania* combination does not disclose, teach, or suggest a system interface operable to "***translate the service request into a non-web service format***," as recited in Claim 45. Thus, Appellants respectfully submit that the rejection of independent Claim 45 is, therefore, improper and should be reversed by the Board.

[1] Webopedia defines SOAP as "a lightweight XML-based messaging protocol used to encode the information in Web service request and response messages before sending them over a network." (See, www.webopedia.com, last visited 10/4/2007).

B. The *Hsu-Catania* combination does not disclose, teach, or suggest a service implementation operable to *"persist the fault . . . wherein persisting the fault comprises attaching a unique identifier to the fault report"*

The rejection of Claim 45 is also deficient at least because the proposed *Hsu-Catania* combination does not disclose, teach, or suggest a service implementation operable to "persist the fault . . . wherein persisting the fault comprises attaching a unique identifier to the fault report," as recited in Claim 45. In the *Final Office Action*, the Examiner relies specifically on *Hsu* for disclosure of the recited claim features. However, the cited portion of *Hsu* merely discloses:

> Exceptions may have associated error data, which may include error codes, stored in the error catalog 210. Some categories of exceptions may not need error data stored in the error catalog 210. For example, exceptions that are concrete subclasses (i.e. a subclass that may have instances or be instantiated rather than inherited) of BusinessException do not need error data in the error catalog, while exceptions that are concrete subclasses of SystemException and FrameworkException do need their error data stored. The error catalog 210 may be loaded based on information in a configuration file or files 212. Among that information are keys used for message display in an error page. When an exception occurs and an action forward is called, the error catalog 210 may be accessed based on the error code and used to determine which error message to display in the resulting page. In some cases, the error code catalog 210 may also be accessed based on the type of error action forward and used to determine the error message to display in the resulting page. With the error code catalog 210, error JSP pages may remain generic. The error message may be determined at runtime and, therefore, may be plugged into the framework of a JSP.

(*Hsu*, column 8, lines 36-54). Thus, although the cited portion discusses the use of error codes, *Hsu* only indicates that the error codes are used to identify the type of message to display and further, indicates that the error codes are applied to "categories of exceptions." Portions of *Hsu* immediately preceding the portion cited by the Examiner clarify that there may be "three separate abstract subclasses" of exceptions and that each exception must fall into one of these categories. (*Hsu*, column 8, lines 18-20). There is no indication that either of the error codes or the error categories are unique to an error instance. Because *Hsu* states that exceptions are classified into categories of exceptions and then only identifies that "error codes" are used to identify the type of error message to display, Appellant respectfully submits that *Hsu* and the proposed *Hsu-Catania* combination do not disclose, teach, or suggest "a service implementation operable to "persist the fault . . . wherein persisting the fault comprises attaching **a unique identifier** to the fault report," as recited in Claim 45.

For at least these additional reasons, Appellants respectfully submit that the rejection of Claim 45 is improper and should be reversed by the Board.

C. The *Hsu-Catania* combination does not disclose, teach, or suggest a fault service implementation operable to *"translate the fault report into a web service format"*

The rejection of Claim 45 is also deficient at least because the proposed *Hsu-Catania* combination does not disclose, teach, or suggest a fault service implementation operable to "translate the fault report into a web service format," as recited in Claim 45. In the *Final Office Action*, the Examiner provides no citation to any specific reference and instead merely states "e.g., WSDL" to reject the recited claim element. (*Final Office Action*, page 12). As such, the *Final Office* Action and the similar communications before it leave Appellants guessing as to which reference the Examiner is relying upon. It is Appellants position, however, that neither *Hsu* nor *Catania* disclose the recited features and operations.

Hsu discloses an "object cache manager 114 enables applications to create customized in-memory cache for storing objects having data originating from backend data stores, such as databases or service based frameworks (e.g., Web Services Description Language "WSDL")." (*Hsu*, Column 6, lines 54-58). Thus, *Hsu* merely indicates that objects originating in WSDL can be stored. To the extent that *Hsu* discloses reporting faults or providing a fault report, *Hsu* only discloses that "the error handler or manager 128 functions to track or chain errors occurring in series, catalog error messages based on error codes, and display error messages using an error catalog." (*Hsu*, column 7, lines 9-12). Appellants find no discussion of **translating fault reports** into a web service format in Hsu. Certainly, the mere disclosure of the tracking, cataloging, and displaying of errors does not include providing a fault service implementation operable to "**translate the fault report into a web service format**," as recited in Appellant's Claim 45.

Catania does not make up for the deficiencies of *Hsu*. As discussed above, *Catania* merely discloses that "[w]eb services typically send XML messages formatted in accordance with the Simple Object Access Protocol (SOAP) specification." (*Catania*, page 1, paragraphs 8). Thus, *Catania* merely discloses that web service requests are sent in the WSDL format.

The service requestor must obtain a copy of the WSDL file from the server and then format the request in the proper SOAP request format prior to it being sent. There is no disclosure of providing a fault service implementation operable to "**translate the fault report into a web service format**," as recited in Appellant's Claim 45.

For at least these additional reasons, Appellants respectfully submit that the rejection of Claim 45 is improper and should be reversed by the Board.

III. Claims 2-5, 11-12, 15-20, 23, 25, and 26 are Allowable over the Proposed *Hsu-Catania* Combination

As one example, independent Claim 2 recites:

> A method for managing faults in a web service architecture comprising:
> receiving a service request in a web service language, wherein the service request comprises invoking a service over a network;
> *translating the service request into a non-web service language*;
> executing the service request;
> encountering an exception during the execution, wherein the execution comprises a fault preventing the fulfillment of the service request;
> persisting the fault; and
> providing a fault response.

Because the proposed *Hsu-Catania* combination does not disclose, teach, or suggest at least the claim elements emphasized above, Appellants submit that Claim 2 is allowable over the proposed *Hsu-Catania* combination. Independent Claims 11 and 26 are allowable for similar reasons.

In the *Final Office Action*, the Examiner relies specifically on *Catania* for disclosure of "translating the service request into a non-web service language," as recited in Claim 2. Appellants respectfully disagree. As discussed above, *Catania* discloses "three primary roles: service provider, service registry, and service requester." (*Catania*, page 1, paragraph 7). "The service provider is the entity that provides access to the Web service and **publishes the service description in a service registry**." (*Catania*, page 1, paragraph 7, emphasis added). By contrast, **"[t]he service requestor finds the service description in a service registry** or other location and can use the information in the description to bind to a service." (*Catania*, page 1, paragraph 7, emphasis added). With regard to the messages that are sent, *Catania* discloses that "[w]eb services typically send XML messages formatted in accordance with the Simple

Object Access Protocol (SOAP) specification." (*Catania*, page 1, paragraphs 8). *Catania* further clarifies:

> The XML messages are described using the Web Services Description Language (WSDL) specification, which, along with the Universal Description Discovery and Integration (UDDI) registry, provides a definition of the interface to a Web service and identifies service providers in a network. The WSDL specification is an XML-based language used to define Web services and describe how to access them. **An application trying to use a particular Web Service can often use WSDL to find the location of the Web service, the function calls available, and the format that must be followed to access the Web service. Therefore, the client first obtains a copy of the WSDL file from the server and then uses the information in this file to format a SOAP request**.

(*Catania*, page 1, paragraphs 8-9, emphasis added). Thus, *Catania* merely discloses that web service requests are sent in the WSDL format. The service requestor must obtain a copy of the WSDL file from the server and then format the request in the proper SOAP request format prior to it being sent. Because the SOAP format is a web service format,[2] Appellant contends that using the WSDL registry to format the request in a SOAP format does not include "translating the service request into a **non-web service language**."

Appellants note the PTO's reliance on paragraphs 68-70 of *Catania* in the *Final Office Action*. (*Final Office Action*, page 3). However, the cited portion of *Catania* merely discloses a distributed business processes example in which an auction manager "offers a management service that monitors the progress of" a request for quotes (RFQ) process 510. (*Catania*, page 5, paragraph 69). "In one embodiment, RFQ process 510 is implemented in the Business Processes Execution Language (BPEL)," which "is an XML-based language designed to enable task sharing for a distributed computing environment, even across multiple organizations, using a combination of Web services. (*Catania*, page 5, paragraph 69). Thus, the service requestor (Companies C2, C3, C4) transmits requests to the web server (auction manager 500) in BPEL. There is no disclosure of the auction manger 5000 "translating the service request into a **non-web service language**," as recited in Claim 2. To the contrary, *Catania* specifically states that the auction manager implements the process in BPEL. Thus, no translation occurs.

[2] Webopedia defines SOAP as "a lightweight XML-based messaging protocol used to encode the information in Web service request and response messages before sending them over a network." (See, www.webopedia.com, last visited 10/4/2007).

For at least these reasons, the *Hsu-Catania* combination does not disclose, teach, or suggest "*translating the service request into a non-web service language*" as recited in Claim 2. Thus, Appellants respectfully submit that the rejections of independent Claim 2 (together with Claims 3-5 that depend on Claim 2) are improper and should be reversed by the Board. Since independent Claims 11 and 26 recite certain similar limitations, Appellants respectfully submit that the rejections of independent Claims 11 and 26 (together with Claims 12, 15-20, 23, and 25 that depend on Claim 11 and Claims 27-38 and 41-44 that depend on Claim 26) are also improper and should be reversed by the Board.

IV. Claims 6, 21-22, and 39-40 are Allowable over the Proposed *Hsu-Catania* Combination

Dependent Claims 6, 21-22, and 39-40 depend upon independent Claims 1, 11, and 26, respectively, and are allowable at least because the proposed *Hsu-Catania* combination does not disclose the combination of claim elements recited in these respective independent claims.

Additionally, Claims 6, 21-22, and 39-40 add additional elements that further distinguish the art. For example, dependent Claim 6 recites that "persisting the fault comprises labeling the fault with a unique identifier." In the *Final Office Action*, the Examiner relies specifically on *Hsu* for disclosure of the recited claim features. (*Final Office Action*, page 6). However, the cited portion of *Hsu* merely discloses:

> . . . [T]he error handler or manager 128 functions to track or chain errors occurring in series, catalog error messages based on error codes, and display error messages using an error catalog. The error catalog of the error manager 128 may enable the use of generic error pages, which the error manager 128 populates with the appropriate error message at run time according to the error catalog.

(*Hsu*, Column 7, lines 9-16). Thus, although *Hsu* discusses the use of error codes, *Hsu* only indicates that error messages are cataloged based on the error codes and that the catalog is used to identify the error message to display. As discussed above, *Hsu* further discloses:

> Exceptions may have associated error data, which may include error codes, stored in the error catalog 210. Some categories of exceptions may not need

> error data stored in the error catalog 210. For example, exceptions that are concrete subclasses (i.e. a subclass that may have instances or be instantiated rather than inherited) of BusinessException do not need error data in the error catalog, while exceptions that are concrete subclasses of SystemException and FrameworkException do need their error data stored. The error catalog 210 may be loaded based on information in a configuration file or files 212. Among that information are keys used for message display in an error page. When an exception occurs and an action forward is called, the error catalog 210 may be accessed based on the error code and used to determine which error message to display in the resulting page. In some cases, the error code catalog 210 may also be accessed based on the type of error action forward and used to determine the error message to display in the resulting page. With the error code catalog 210, error JSP pages may remain generic. The error message may be determined at runtime and, therefore, may be plugged into the framework of a JSP.

(*Hsu*, column 8, lines 36-54). Thus, *Hsu* again only indicates that the error codes are used to identify the type of message to display and further, indicates that the error codes are applied to "categories of exceptions." Portions of *Hsu* immediately preceding clarify that there may be "three separate abstract subclasses" of exceptions and that each exception must fall into one of these categories. (*Hsu*, column 8, lines 18-20). There is no indication that either of the error codes or the error categories are unique to an error instance. Because *Hsu* states that exceptions are classified into categories of exceptions and then only identifies that "error codes" are used to identify the type of error message to display, Appellant respectfully submits that *Hsu* and the proposed *Hsu-Catania* combination do not disclose, teach, or suggest that "persisting the fault comprises labeling the fault with **a unique identifier**," as recited in Claim 6.

For at least these additional reasons, Appellants respectfully submit that the rejection of Claim 6 is improper and should be reversed by the Board. Since Claims 21-22 and 39-40 recite certain similar limitations, Appellants respectfully submit that the rejections of Claims 21-22 and 39-40 are also improper and should be reversed by the Board.

V. Claim 24 is Allowable over the Proposed *Hsu-Catania* Combination

Dependent Claim 24 depends upon independent Claim 1 and is allowable at least because the proposed *Hsu-Catania* combination does not disclose the combination of claim elements recited in Claim 1.

Additionally, Claim 24 adds additional elements that further distinguish the art. For example, dependent Claim 24 recites that "the fault service implementation is further operable to translate the fault information into a web service language." In the *Final Office Action*, the Examiner relies specifically on *Hsu* for disclosure of the recited claim features. (*Final Office Action*, page 7). However, the cited portion of *Hsu* merely discloses:

> The object cache manager 114 enables applications to create customized in-memory cache for storing objects having data originating from backend data stores, such as databases or service based frameworks (e.g., Web Services Description Language "WSDL").

(*Hsu*, Column 6, lines 54-58). Thus, the cited portion of *Hsu* merely indicates that objects originating in WSDL can be stored. To the extent that *Hsu* discloses reporting faults or providing a fault report, *Hsu* only discloses that "the error handler or manager 128 functions to track or chain errors occurring in series, catalog error messages based on error codes, and display error messages using an error catalog." (*Hsu*, column 7, lines 9-12). Appellants find no discussion of **translating fault reports** into a web service format in Hsu. Certainly, the mere disclosure of storing WSDL objects and the mere disclosure of tracking, cataloging, and displaying of errors does not include a "fault service implementation is further operable to translate the fault information into a web service language," as recited in Appellant's Claim 24.

For at least these additional reasons, Appellants respectfully submit that the rejection of Claim 24 is improper and should be reversed by the Board.

VI. The Proposed *Hsu-Catania* Combination is Improper

Appellants further submit that one of ordinary skill in the art at the time of Appellants' invention would not have been motivated to make the proposed *Hsu-Catania* combination. According to the M.P.E.P., in order "[t]o establish a prima facie case of obviousness . . . there must be some suggestion or motivation either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings." See §2143. Notably absent from this list is an allowance for an Examiner's conjectured assertion as to the motivation to combine reference teachings. A quote from the M.P.E.P. is directed specifically to this point: "[t]he examiner and the board asserted that it would have been within the skill of the art to substitute one type of detector for another in the system of the primary reference, however the court found there was **no support or explanation** of this conclusion and reversed." See §2143.01 (I) (emphasis added). Thus, "the proper inquiry is whether there is something in the prior art as a whole to suggest the *desirability* . . . of making the combination." *Id.* (internal quotations omitted) (emphasis original). "The mere fact that the references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the **desirability** of the combination." See §2143.01 (III) (emphasis original and added). Most recently, this requirement has been reaffirmed in an official USPTO memorandum dated May 3, 2007 wherein the Deputy Commissioner for Patent Operations pointed to sections of *KSR v. Teleflex*, which recite, "it will be necessary . . . to determine whether there was an **apparent reason** to combine the known elements in the fashion claimed by the patent at issue."[3]

In the *Final Office Action*, the Examiner states "it would thus be obvious to one of ordinary skill in the art at the time of the invention to combine and/or modify *Hsu's* invention with the added feature of the method further comprising translating the service request into a non-web service language, as disclosed by *Catania*, for the motivation of providing a system and method that allows a manager to subscribe to selected to types of events and notify the at least one manage of the occurrence of the selected types of events [Abstract]." (*Final Office Action*, page 5). Appellants respectfully disagree.

Catania makes clear that to the extent that any translation is performed it is done by the client before the request is sent. Specifically, *Catania* states:

> An application trying to use a particular Web Service can often use WSDL to find the location of the Web service, the function calls available, and the format that must be followed to access the Web service. Therefore, the client first obtains a copy of the WSDL file from the server and then uses the information in this file to format a SOAP request.

(*Catania*, page 1, paragraphs 8-9, emphasis added). Thus, *Catania* requires that service requestor obtain a copy of the WSDL file from the server and format the request in the proper SOAP request format prior to it being sent. Accordingly, were one of ordinary skill to combine the disclosure of *Catania* with *Hsu*, the resulting system would require that the requestor format the request in the proper SOAP request before it is sent. The proposed

[3] *KSR Int'l. Co v. Teleflex Inc.*, No. 04-1350 (April 30, 2007) (emphasis added).

combination would not include receiving a service request in a web service language and then translating the service request into a non-web service language. Further modification of *Catania* to result in the service requests being formatted after transmission from the requestor would change the principle of operation disclosed in *Catania*, and the M.P.E.P. makes clear that if a "proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious." (M.P.E.P. § 2143.01).

Accordingly, the Office Action has not shown "something in the prior art as a whole to suggest the *desirability*" of combining *Hsu* with *Catania*, but rather seems to rely on a conjectured assertion that "the references <u>can</u> be combined" without regard to the "desirability of the combination." This directly conflicts with the M.P.E.P. requirements for supporting a motivation to combine references. Appellant therefore further requests that the rejections of the present claims be withdrawn for want of a *prima facie* showing of obviousness as defined by the M.P.E.P. and for the various other reasons described above.

CONCLUSION

Appellant has demonstrated that the present invention, as claimed, is clearly distinguishable over the prior art cited by the Examiner. Therefore, Appellant respectfully requests the Board to reverse the final rejections and instruct the Examiner to issue a Notice of Allowance with respect to all pending claims.

The Commissioner is hereby authorized to charge $540.00 for filing this Brief in support of an Appeal to Deposit Account No. 05-0765 of Electronic Data Systems Corporation. No other fees are believed due; however, the Commissioner is authorized to charge any additional fees or credits to Deposit Account No. 05-0765 of Electronic Data Systems Corporation.

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Appellant

Jenni R. Moen

Jenni R. Moen
Reg. No. 52,038
(214) 953-6809

Dated: February 23, 2009

Correspondence Address:

at Customer No. **35005**

APPENDIX A

Pending Claims

2. A method for managing faults in a web service architecture comprising:

receiving a service request in a web service language, wherein the service request comprises invoking a service over a network;

translating the service request into a non-web service language;

executing the service request;

encountering an exception during the execution, wherein the execution comprises a fault preventing the fulfillment of the service request;

persisting the fault; and

providing a fault response.

3. The method of Claim 2, wherein the service request is received from a service consumer, the service consumer coupled to the network.

4. The method of Claim 3, wherein the fault response is provided to a fault service consumer, and wherein the fault service consumer is coupled to the network.

5. The method of Claim 4, wherein the fault service consumer is the same as the service consumer.

6. The method of Claim 2, wherein persisting the fault comprises labeling the fault with a unique identifier.

7. The method of Claim 6, further comprising storing the fault in a database.

8. The method of Claim 7, further comprising storing multiple faults in the database, the storage comprising storing fault information.

9. The method of Claim 8, wherein providing a fault response comprises providing access to the database, the access operable to permit a user to track any fault stored in the database.

10. The method of Claim 8, wherein providing a fault response further comprises presenting the fault information in a console, the console operable to list the fault information stored in the database.

11. A system for managing faults in a service-oriented architecture comprising:

a service interface operable to:

receive a service request via a network, the service request received in a web service language; and

translate the service request into a non-web service language;

a service implementation coupled to the service interface, the service implementation operable to perform the service request and determine the source of any fault encountered in the performance;

a persistent store operable to persist any faults encountered in the performance; and

a fault service interface operable to transmit fault information.

12. The system of Claim 11, further comprising a fault service implementation coupled to the fault service interface, the fault service interface operable to retrieve the fault information from the persistent store.

15. The system of Claim 12, wherein the fault service interface is further operable to receive fault status requests in a web service language and translate the fault status request into a non-web service language.

16. The system of Claim 11, further comprising a service consumer, the service consumer coupled to the network and operable to transmit the service request to the service interface.

17. The system of Claim 11, further comprising a fault service consumer, the fault service consumer coupled to the network and operable to receive the fault information from the fault service interface.

18. The system of Claim 17, wherein the fault service consumer and the service consumer are the same consumer.

19. The system of Claim 12, further comprising a fault network coupled to the network, the fault network operable to couple the service interface, service implementation, persistent store, and fault service interface.

20. The system of Claim 11, wherein the persistent store is a database operable to store faults encountered during the performance.

21. The system of Claim 11, wherein the service implementation is further operable to attach a unique identifier to each fault.

22. The system of Claim 21, wherein the service implementation is further operable to direct the persistent store to store any faults according to the unique identifier.

23. The system of Claim 20, wherein the database is further operable to store the faults in a web service language.

24. The system of Claim 12, wherein the fault service implementation is further operable to translate the fault information into a web service language.

25. The system of Claim 12, further comprising a console, the console operable to display fault information retrieved by the fault service implementation.

26. A system for managing faults in a web service architecture comprising:

a web service module coupled to a network and operable to manage service requests in a web service language, the web service module operable to:

receive a service request via a network, the service request received in the web service language; and

translate the service request into a non-web service language;

a diagnostic module operable to fulfill the service request and identify faults associated with the service request; and

a fault persistence module operable to store the faults in a persistent store.

27. The system of Claim 26, wherein the web service language is any protocol registered in the Universal Description Discovery and Integration registry.

28. The system of Claim 26, wherein the web service language is a remote procedure call.

29. The system of Claim 26, wherein the web service language is a HyperText Transfer Protocol.

30. The system of Claim 26, wherein the web service language is an application service interface.

31. The system of Claim 30, wherein the application service interface is Java message service.

32. The system of Claim 26, wherein the web service language is a protocol approved as a web service description language approved by the World Wide Web Consortium.

33. The system of Claim 26, wherein the persistent store is a database dedicated to the fault persistence module.

34. The system of Claim 26, wherein the web service module is further operable to receive service requests.

35. The system of Claim 26, further comprising a sub-network coupled to the web services module.

36. The system of Claim 35, further comprising at least one internal system, the at least one internal system coupled to the sub-network and operable to provide information required by the service request.

37. The system of Claim 36, wherein the diagnostic module is further operable to identify any faults caused by the at least one internal system.

38. The system of Claim 37, wherein the diagnostic module is further operable to communicate any faults to the fault persistence module.

39. The system of Claim 38, wherein the fault persistence module is further operable to label each fault with a unique identifier.

40. The system of Claim 39, wherein the fault persistence module is further operable to direct the persistent store to organize each fault by a unique identifier.

41. The system of Claim 26, wherein the web service module is further operable to receive a fault status request.

42. The system of Claim 41, wherein the fault status request is sent by a fault service consumer.

43. The system of Claim 42, wherein the fault service consumer is coupled to the sub-network.

44. The system of Claim 42, wherein the fault service consumer and the service consumer are the same.

45. A system for managing faults in a web services architecture comprising:

a system interface operable to receive a service request in a web services format, the system interface further operable to translate the service request into a non-web service format;

a service implementation operable to fulfill the service request, generate a fault report, and persist the fault, the persistence comprising storing the fault report in a persistent store, wherein generating a fault report comprises detecting a fault during the fulfillment of the service request, and persisting the fault comprises attaching a unique identifier to the fault report;

a fault service implementation operable to retrieve the fault report from the persistent store and translate the fault report into a web service format; and

a fault service interface operable to receive fault service requests and transmit a fault service response.

APPENDIX B

Evidence Appendix

Other than the references attached to this Appeal Brief as Appendices A and B, no evidence was submitted pursuant to 37 C.F.R. §§ 1.130, 1.131, or 1.132, and no other evidence was entered by the Examiner and relied upon by Appellant in the Appeal.

APPENDIX C

Related Proceedings Appendix

As stated on Page 3 of this Appeal Brief, to the knowledge of Appellant's Counsel, there are no known appeals, interferences, or judicial proceedings that will directly affect or be directly affected by or have a bearing on the Board's decision regarding this Appeal.